"To explain the integration of information, we need only exhibit mechanisms by which information is brought together and exploited by later processes."   
  
-David Chalmers, Facing Up to The Problem of Consciousness

# Introduction

## Objective

One defining aspect of human intelligence is our ability to subconsciously form new connections between abstract concepts, which then seem to "bubble up" to the forefront of our attention. This phenomenon, commonly called intuition, is responsible not only for our most startling and profound "Aha!" moments, but also for the seemingly arbitrary changes in our awareness of, say, the ticking of a clock on the wall. And although intuition is, unfortunately, a system that exists inside us as a “black box” (we have no conscious access to its decision-making process), the realization that we experience these shifts of attention unwillingly and "out of the blue" provides powerful clues to its underlying mechanisms.

The primary purpose of this project then, was to develop a framework for exploring these ideas as well as write and train an ensemble learning system (an *agent*) who's “conscious awareness” (its *attention*) receives input as connections between the symbolic elements of its environment as served up by its *intuition*. Further, noting the hierarchal nature of information, the agent was to be designed as one possible node in a larger network of such agents, facilitating the bootstrapping of an increasingly advanced intuition.

## Technologies

The framework and agent were developed in the Python (3.6) programming language. 3rd party libraries include PyTorch (machine learning) and KarooGP (genetic programming).

# Problem Domain

A problem domain to facilitate exploration of this topic was chosen based on the criteria that the agent’s environment should, ideally:

* Afford the agent an opportunity to explore a complex, unknown search-space.
* Be multi-context.
* Provide mechanisms for signaling feedback to the agent.
* Have the potential for practical application.

With that in mind, the agent was applied to the task of learning the Python programming language and, eventually, accomplishing some goal using it. This meets our criteria in the following way -

* The Python programing space is complex, and an isolated search space may be constructed.
* Various contexts exist, such as keywords, functions, arguments, classes, instances, programs etc.
* The Python standard library exposes methods such as keyword.iskeyword(s) and callable(s), enabling the agent to “ask” if a string s represents a Python keyword or a callable function. Further, because the agent itself is written in Python, custom methods may be written to provide more specific feedback, such as Connector.is\_python(s), which returns true if s represents a valid Python program with no syntax errors and generating no exceptions.
* Because of Python’s capability for reflection, a sufficiently advanced agent might write a version of itself to solve some arbitrary problem (indeed, even writing its own feedback mechanisms, in Python, for signaling fitness heuristics to itself.)

# Modeling Intuition

## The Model

To build an agent with intuition, a model was developed based on the following observations of human behavior

* Humans must often react to an event faster than they have time to logically come to an informed decision about them (CITE). However, we are able make very good “in-the-moment” approximations that seem to be automatic/subconscious.
* Intuition must have a Darwinian utility function, something like “is our current contextual awareness and behavior contributing to homeo-stasis, or not?”
* Shifts of attention from one context to the next are often (if not always) autonomic.
* It is likely heuristics appear to guide it.
* It is in our nature to explore ourselves and our origins.
* Complex systems in nature are feedback networks (CITE), and the brain and its myriad sub-systems appear to be no exception. With that in mind, the following model was developed
* It is likely a feedback network receiving it’s signals from both the current environmental context and our awareness of, our “concentration on”, those signals, as it seems to guide us as much as we guide it.
* we don’t get to decide which signals it gives us, and it seems to have access to our sensory-input more holistically, where our attention tends to have tunnel-vision and miss important environmental queues (at least until our sub-conscious intuition serves them up as pertinent, that is).

The model conceived consists of 3 layers, labeled *Conceptual, Intuitive,* and *Attentive.* As described below and given by Figure 1.

Data is mostly feed-forward, with recurrent feedback signaling the agent's current state and contextual fitness.

FROM INTRO: an ensemble learning system (an *agent*) who's “conscious awareness” (its *attention*) receives input as connections between the symbolic elements of its environment according to its *intuition*.

**The agent/model are based on the following observations**

The model was constructed to represent the following

* With A arranged in this way, each ANN's output is dependent on current sensory input as well as the agent's current context.
* Intuition is not ephemeral
* How is the genetic algorithm diff than just naother ann? Does it prevent overtraining
* How to simulate a positve change in the agent's environment - alter input/less noise? "Feed in what you know" (ex: concentrating)?
* The agent exists in the context of its inputs/environment and intuitively learns the symbols present in it - it might be thought of as an automatic tokenizer.
* Purpose of "mistakes" is a signal to "check in" on this process or connectoin or whatever it represents.
* The difference between this model and a single input "line" is the extensibility of memory and introduction of error into the connection-forming process.
* "Lottery scheduling", but in an evolving way - we learn an optimal schedule according to what is in our current awareness vs what isn't - it can be thought of as a temporal priroirty
* Intution vs analytical system - genetic alg decides allocation between two systems.
* We can only focus on one context at a time, how do we weight them in order to make a decision?
* How are the inputs weighted? Biased by "amount" of input and log(type)? I.e. Fire vs. TV - Fire is big and new.
* Context - an amalgamation of input from the five-senses input - we listen and see at the same time?

### Layer 1

The conceptual layer represents one's existing knowledge base (KB) of abstract concepts. It consists of a set `A` of artificial neural networks (ANNs) with the following properties -

Each ANN at this level has a set `X` of input nodes consisting of `k` feed-forward sensory input nodes and `m` feedback input nodes, defined as:

| X | Channel | Description |

|-------------|-------------| -------------|

| `x\_0` to `x\_k-1` | Feed-forward | Environmental sensory input, identical for every `a` in `A` |

| `x\_k` to `x\_k+m-1` | Feedback | Each `x\_k` to `x\_n` is mapped from a corresponding attentive-layer input node. This represents feedback of the agent's current context. |

Each ANN is pre-trained (offline), possibly each for a different class of objects (ex: `a\_0` classifies digits, `a\_1` classifies letters, etc). During this pre-training, each `x\_k` to `x\_m-1` input-value should be randomized to simulate environmental noise.

Each ANN's output nodes are provided to the intuitive layer as input.

### Layer 2

The intuitive layer is a set of data pipes, one for each conceptual-layer ANN, connecting the conceptual layer to the attentive layer. On each state change, each pipe is weighted according to some fitness function that evolves in an online manner according to some genetic algorithm, who's fitness is received as feedback from the attentive layer.

These weights are subsequently used as a bias (possibly binary, logarithmic, etc) by the attentive layer. In this way, the agent's "intuition" learns how to best allocate the agent's "attention" while allowing "mistakes" to enter its awareness. These mistakes represent possible new connections between the conceptual layer's existing abstract concepts.

### Layer 3 – The Logical Layer

The logical layer examines the output-nodes of the intutive layer to draw concl

## Heuristics

* What heuristics might drive it and its exploration - neophilia? Self-actualization? Guilt?
* Our current awareness dictates “work” done by subconscious
* How are the inputs weighted? Biased by "amount" of input and log(type)? I.e. Fire vs. TV - Fire is big and new.
* Context - an amalgamation of input from the five-senses input - we listen and see at the same time?

# The Data

Extracted w/Pandas

Why this set? Familiar with it. Deep learnable. No conv or pool layers

Data Set desc: <https://archive.ics.uci.edu/ml/datasets/Letter+Recognition>

## The Agent

LEARNING\_ITERATIONS

LEARNING\_RATE

BIAS

BIAS\_WEIGHT

NETWORK\_LAYERS

NETWORK\_LAYER\_COUNT

TRAINING\_DATAFILE

VALIDATION\_DATAFILE

# Performance

* Human context switching is approx 200 ms. 20-50ms is reasonably real-time.

# Failed Approaches/Models

…

# What’s Next

Bootstrapping:

Show how one node has one goal, a clique has another goal, a network has an even still more encompassing goal (ex: find free memory?)

L2 Tuning:

1. Monitor accuracy heuristics over time -

1. increase max pop size after some accuracy threshold

3. decrease max pop size if proprtion of unfit to fit outputs calls for it

Branching:

1. Branch into a hierarichal structure of agents (faciliated by reflection)

2. L2 branch after some number of increases of the last x iters

3. Each agent represents a single "concept" such as a letter, or a python kwd

4. If a branch agent does not learn enough over some t, rm it (log to kb?) - it's inputs do not form any concepts

Other heuristics:

# Influences

Nicolai Tesla, Daniel Hofstadter, Ray Kurzweil, Richard Dawkins, Jacques Pitrat.

# Appendix A: Technical Information

# Technologies / 3rd Party Libraries

* Python: reflection, etc.
* (From Technologies section) The agent is developed in Python using the PyTorch and KarooGP machine learning and genetic programming libraries, respectively.
* Python was chosen for its rapid prototyping capabilities and ability for reflection.

# Usage Instructions

…